PDR RID Report

Date Last Modified 6/8/95
Originator Groff, Robert; Isaac David **Phone No** 301-901-9236
Organization MITRE
E Mail Address rgroff@mitre.org
Document DID 305 Design Spec
Section Section 6.3.3 **Page** NA **Figure Table** NA

RID ID PDR 312
Review CSMS
Originator Ref Design 2.17
Priority 2

Category Name Design-CSS **Actionee** HAIS
Sub Category
Subject Discuss approach for controlling Thread Service limitations

Description of Problem or Suggestion:

The text states that when setting up a server you need to "Specify the maximum number of threads the service can run in order to execute the user specified services concurrently." What is the practical (performance and other issues considered) limit on the number of concurrent threads? Can this limit be exceeded during realistic usage of ECS? If this limit can be realistically exceeded, what approach will be used for dealing with this problem?

Originator's Recommendation

Address the issue. Discuss the approach to be used when the number of concurrent threads is exceeded.

GSFC Response by: **GSFC Response Date**

HAIS Response by: Winston **HAIS Schedule**
HAIS R. E. Hota **HAIS Response Date** 5/2/95

As part of the Developers Guidelines (to be provided by CDR), directives will be provided to the applications developers to help them establish an optimum number of threads that a particular server should limit itself to use.

While there is effectively no limitation in UNIX as to how many threads may actually by spawned, there is an effective processing limit on each platform. This effective processing limit exists whether or not threads are used. One cannot with authority state what the effective (processing-wise) thread limitation is unless one can specify the processing requirements each thread is being programmed to handle. One can observe however, that each thread does create a certain amount of work (overhead) for the operating system to handle (whether it is in the time-slice or the queuing mechanism). During operations, if an effective processing limit is being reached by a server or by a platform, monitoring is in place to identify this condition and operational proceedures will be in place to manage this situation.

Threads are considered a light-weight process in UNIX and as such have little overhead compared to a separate process running in UNIX. There is no limitation as to how many threads a developer can specify in a server to receive incoming calls. It is effectively the amount of work that is to be carried out in each thread that could effect the performance of the UNIX host. [This is an implementation issue on assigning the number of hosts and the numbers of a particular server that will be required to perform the anticipated workload.] While assigning time slices to processes, the operating system (scheduler) treats each thread as if it is a separate process and assigns a time slice for each thread. As such, performance as a whole depends on the host CPU (except for a little overhead due to threads). When all the specified number of threads are running (executing incoming calls), other calls are kept in a queue whose normal default size is equal to eight times the number of threads the service can handle simultaneously. This default can be over ridden by specifying a larger number if the developer foresees a need. The service rejects any calls that it receives if the queue is filled up. The caller should check the return status and try to invoke the service at a later time should the invocation fail. Another implementation alternative is to run more than one application server (either on the same host or on different hosts) so the calling application can then(if after failing to reach a given service) bind to a different service (with the same interface) and invoke the service.

Status **Closed** **Date Closed** **6/8/95** **Sponsor** **Broder**

****** **Attachment if any** ******

PDR RID Report

****** Attachment if any ******